# Validation par analyse statique

## Interprétation abstraite - Deuxième partie

Pierre-Loïc Garoche

ONERA

Cours ISAE
2014-2015

Slides majoritairement empruntés au docteur Pierre Roux



## Sémantique abstraite - suite





## Sémantique abstraite - suite





## Analyse en arrière

On avait défini la sémantique abstraite des gardes comme

$$[\![e > 0]\!]^\sharp_{\mathrm{C}}\ \rho = \begin{cases} \rho\left[v \mapsto \rho(v)\ \sqcap^\sharp \alpha\left([\![1, +\infty[\![\right)\right] & \text{si } e = v \\ \rho & \text{sinon} \end{cases}$$

Comment faire pour $x - 4 > 0$ ?

## Analyse en arrière

On avait défini la sémantique abstraite des gardes comme

$$[\![ e > 0 ]\!]^\sharp_{\mathrm{C}}\ \rho = \begin{cases} \rho\left[v \mapsto \rho(v)\ \sqcap^\sharp \alpha\left([\![1, +\infty[\![\right)\right] & \text{si } e = v \\ \rho & \text{sinon} \end{cases}$$

Comment faire pour $x - 4 > 0$ ?

On va utiliser une analyse en arrière des expressions :
partant du résultat de l'expression,
on en déduit les valeurs possibles des variables.



## Analyse en arrière (suite)

Sémantique en arrière des expressions :
$[\![e]\!]\!\downarrow^\sharp : (\mathbb{V} \to \mathcal{D}^\sharp) \times \mathcal{D}^\sharp \to (\mathbb{V} \to \mathcal{D}^\sharp)$



## Analyse en arrière (suite)

Sémantique en arrière des expressions :
$[\![e]\!]\!\downarrow^\sharp : (\mathbb{V} \to \mathcal{D}^\sharp) \times \mathcal{D}^\sharp \to (\mathbb{V} \to \mathcal{D}^\sharp)$

$$[\![v]\!]\!\downarrow^\sharp(\rho, r) \qquad = \rho\,[v \mapsto \rho(v) \sqcap r]\,(v)$$



## Analyse en arrière (suite)

Sémantique en arrière des expressions :
$[\![e]\!]\!\downarrow^\sharp : (\mathbb{V} \to \mathcal{D}^\sharp) \times \mathcal{D}^\sharp \to (\mathbb{V} \to \mathcal{D}^\sharp)$

$$[\![v]\!]\!\downarrow^\sharp(\rho, r) \qquad = \rho\,[v \mapsto \rho(v) \sqcap r]\,(v)$$

$$[\![n]\!]\!\downarrow^\sharp(\rho, r) \qquad = \begin{cases} \bot & \text{si } n^\sharp \sqcap^\sharp r = \bot \\ \rho & \text{sinon} \end{cases}$$

## Analyse en arrière (suite)

Sémantique en arrière des expressions :
$[\![e]\!]\!\downarrow^\sharp : (\mathbb{V} \to \mathcal{D}^\sharp) \times \mathcal{D}^\sharp \to (\mathbb{V} \to \mathcal{D}^\sharp)$

$$[\![v]\!]\!\downarrow^\sharp(\rho, r) = \rho\,[v \mapsto \rho(v) \sqcap r]\,(v)$$

$$[\![n]\!]\!\downarrow^\sharp(\rho, r) = \begin{cases} \perp & \text{si } n^\sharp \sqcap^\sharp r = \perp \\ \rho & \text{sinon} \end{cases}$$

$$[\![\mathbf{rand}(n_1, n_2)]\!]\!\downarrow^\sharp(\rho, r) = \begin{cases} \perp & \text{si } \mathbf{rand}^\sharp(n_1, n_2) \sqcap^\sharp r = \perp \\ \rho & \text{sinon} \end{cases}$$



## Analyse en arrière (suite)

Sémantique en arrière des expressions :
$[\![e]\!]\!\downarrow^\sharp : (\mathbb{V} \to \mathcal{D}^\sharp) \times \mathcal{D}^\sharp \to (\mathbb{V} \to \mathcal{D}^\sharp)$

$$[\![v]\!]\!\downarrow^\sharp(\rho, r) = \rho\,[v \mapsto \rho(v) \sqcap r]\,(v)$$

$$[\![n]\!]\!\downarrow^\sharp(\rho, r) = \begin{cases} \perp & \text{si } n^\sharp \sqcap^\sharp r = \perp \\ \rho & \text{sinon} \end{cases}$$

$$[\![\mathbf{rand}(n_1, n_2)]\!]\!\downarrow^\sharp(\rho, r) = \begin{cases} \perp & \text{si } \mathbf{rand}^\sharp(n_1, n_2) \sqcap^\sharp r = \perp \\ \rho & \text{sinon} \end{cases}$$

$$[\![e_1 + e_2]\!]\!\downarrow^\sharp(\rho, r) = [\![e_1]\!]\!\downarrow^\sharp(\rho, r_1) \sqcap^\sharp_{\mathrm{nr}} [\![e_2]\!]\!\downarrow^\sharp(\rho, r_2)$$
$$\text{avec } (r_1, r_2) = +\!\downarrow^\sharp\left([\![e_1]\!]^\sharp_{\mathrm{E}}(\rho), [\![e_2]\!]^\sharp_{\mathrm{E}}(\rho), r\right)$$

...



## Analyse en arrière, arithmétique

**Exemple**

Dans le domaine des signes :
$+\!\downarrow^\sharp(\geqslant 0, \geqslant 0, \leqslant 0) = (0, 0)$



## Analyse en arrière, arithmétique

**Exemple**

Dans le domaine des signes :
$+\!\downarrow^\sharp(\geqslant 0, \geqslant 0, \leqslant 0) = (0, 0)$

(si $x \geqslant 0,\ y \geqslant 0$ et $x + y \leqslant 0$ alors $x = y = 0$)

## Analyse en arrière, arithmétique

**Exemple**

Dans le domaine des signes :
$+\downarrow^\sharp(\geqslant 0, \geqslant 0, \leqslant 0) = (0, 0)$
(si $x \geqslant 0$, $y \geqslant 0$ et $x + y \leqslant 0$ alors $x = y = 0$)

**Exemple**

Dans le domaine des intervalles :
$+\downarrow^\sharp(\llbracket 0, 2\rrbracket, \llbracket 3, 8\rrbracket, \llbracket 4, 7\rrbracket) = (\llbracket 0, 2\rrbracket, \llbracket 3, 7\rrbracket)$



## Analyse en arrière, arithmétique

**Exemple**

Dans le domaine des signes :
$+\downarrow^\sharp(\geqslant 0, \geqslant 0, \leqslant 0) = (0, 0)$
(si $x \geqslant 0$, $y \geqslant 0$ et $x + y \leqslant 0$ alors $x = y = 0$)

**Exemple**

Dans le domaine des intervalles :
$+\downarrow^\sharp(\llbracket 0, 2\rrbracket, \llbracket 3, 8\rrbracket, \llbracket 4, 7\rrbracket) = (\llbracket 0, 2\rrbracket, \llbracket 3, 7\rrbracket)$

**Exercices**

- Donner la table de $+\downarrow^\sharp$ pour le domaine des signes (tout au moins une partie, la table ayant 125 entrées).



## Analyse en arrière, arithmétique

**Exemple**

Dans le domaine des signes :
$+\downarrow^\sharp(\geqslant 0, \geqslant 0, \leqslant 0) = (0, 0)$
(si $x \geqslant 0$, $y \geqslant 0$ et $x + y \leqslant 0$ alors $x = y = 0$)

**Exemple**

Dans le domaine des intervalles :
$+\downarrow^\sharp(\llbracket 0, 2\rrbracket, \llbracket 3, 8\rrbracket, \llbracket 4, 7\rrbracket) = (\llbracket 0, 2\rrbracket, \llbracket 3, 7\rrbracket)$

**Exercices**

- Donner la table de $+\downarrow^\sharp$ pour le domaine des signes (tout au moins une partie, la table ayant 125 entrées).
- Définir $-\downarrow^\sharp$ pour le domaine des intervalles.



## Analyse en arrière, arithmétique (suite et fin)

**Réponse**

$-\downarrow^\sharp(\llbracket a, b\rrbracket, \llbracket c, d\rrbracket, \llbracket e, f\rrbracket) = (\llbracket \max(a, e + c), min(b, f + d)\rrbracket,$
$\llbracket \max(c, a - f), \min(d, b - e)\rrbracket)$

$-\downarrow^\sharp(x^\sharp, y^\sharp, r^\sharp) = (\perp, \perp)$ sinon (si $x^\sharp$, $y^\sharp$ ou $z^\sharp$ est $\perp$).

## Exercice, analyse en arrière (suite et fin)

**Exercice**

- Avec la sémantique en arrière des expressions, définir une sémantique abstraite pour les gardes plus précise.
- Puis calculer cette sémantique dans le domaine des intervalles pour la garde $x + y \leqslant z$ avec $\rho(x) = [\![1, 10]\!]$, $\rho(y) = [\![3, 10]\!]$ et $\rho(z) = [\![3, 5]\!]$.



## Exercice, analyse en arrière (suite et fin)

**Exercice**

- Avec la sémantique en arrière des expressions, définir une sémantique abstraite pour les gardes plus précise.
- Puis calculer cette sémantique dans le domaine des intervalles pour la garde $x + y \leqslant z$ avec $\rho(x) = [\![1, 10]\!]$, $\rho(y) = [\![3, 10]\!]$ et $\rho(z) = [\![3, 5]\!]$.

**Réponse**

- $[\![e > 0]\!]^\sharp_C\ \rho = [\![e]\!]\!\downarrow^\sharp(\rho, \alpha([\![1, +\infty[\![))$



## Exercice, analyse en arrière (suite et fin)

**Exercice**

- Avec la sémantique en arrière des expressions, définir une sémantique abstraite pour les gardes plus précise.
- Puis calculer cette sémantique dans le domaine des intervalles pour la garde $x + y \leqslant z$ avec $\rho(x) = [\![1, 10]\!]$, $\rho(y) = [\![3, 10]\!]$ et $\rho(z) = [\![3, 5]\!]$.

**Réponse**

- $[\![e > 0]\!]^\sharp_C\ \rho = [\![e]\!]\!\downarrow^\sharp(\rho, \alpha([\![1, +\infty[\![))$
- On obtient : $\rho(x) = [\![1, 3]\!]$, $\rho(y) = [\![2, 5]\!]$ et $\rho(z) = [\![4, 5]\!]$.



## Exercice : domaine des congruences

**Exercice**

- Concevoir un domaine abstrait non relationnel pour les congruences (exemple : $x$ est congru à 2 modulo 4 : $x \in 4\mathbb{Z} + 2$).
- Analyser avec le programme du premier exemple :

```
x = rand(0, 12) ; y = 42 ;
while (x > 0) {
      x = x − 2 ;
      y = y + 4 ;
}
```

Sémantique abstraite - suite

Abstractions relationnelles

- Rappel
- Polyèdres
- Octogones

Contrôleurs d'avion

Si le cours avait duré un semestre...

- Domaines non numériques
- Virgule flottante
- Partitionnement
- Stratégies d'itération

Outils existants



Sémantique abstraite - suite

Abstractions relationnelles

- Rappel
- Polyèdres
- Octogones

Contrôleurs d'avion

Si le cours avait duré un semestre...

- Domaines non numériques
- Virgule flottante
- Partitionnement
- Stratégies d'itération

Outils existants



## Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes



## Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles

## Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes
- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles
  - $+$ plus précis
  - $-$ plus compliqué et plus coûteux



## Comment abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ ?

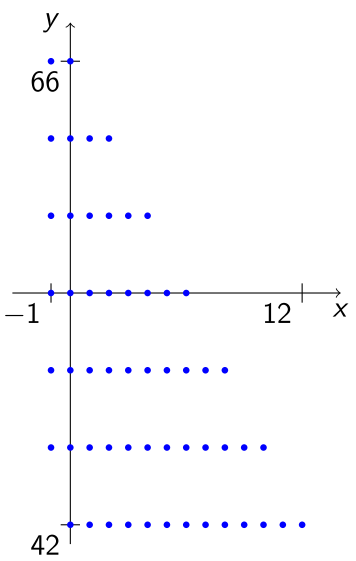
Deux grandes solutions

- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ en $\mathbb{V} \to \mathcal{P}(\mathbb{Z})$ puis $\mathcal{P}(\mathbb{Z})$ en un $\mathcal{D}^\sharp$
  - *non relationnel* : les valeurs de $x$ et $y$ sont indépendantes
  - la semaine dernière
- Abstraire $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$ directement en un $\mathcal{D}^\sharp$
  - *relationnel* : certaines combinaisons de $x$ et $y$ sont impossibles
  - $+$ plus précis
  - $-$ plus compliqué et plus coûteux
  - cette semaine



## Deux petits dessins valent mieux que de longs discours

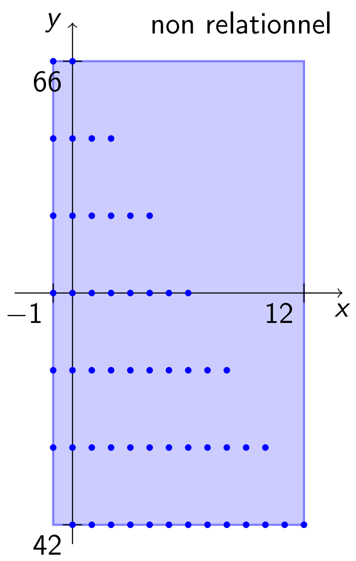
Exemple précédent au point de programme 2 (invariant de boucle)





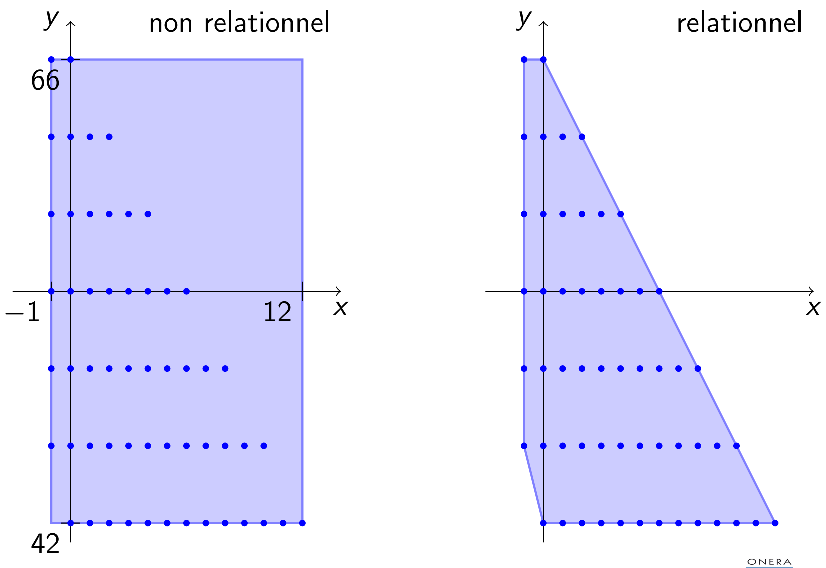

## Deux petits dessins valent mieux que de longs discours

Exemple précédent au point de programme 2 (invariant de boucle)



## Deux petits dessins valent mieux que de longs discours

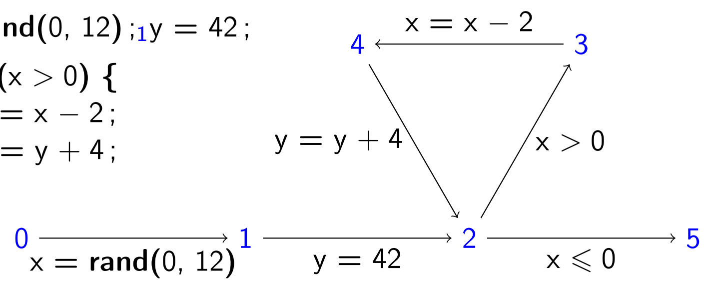
Exemple précédent au point de programme 2 (invariant de boucle)






## Limitations des domaines non relationnels

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) **{**
  $_3$x = x − 2 ;
  $_4$y = y + 4 ;
**}**$_5$




- Pour borner $y$, on a besoin de l'invariant $2x + y \leqslant 66$.
- Cet invariant de boucle ne peut être exprimé par aucun domaine non relationnel.





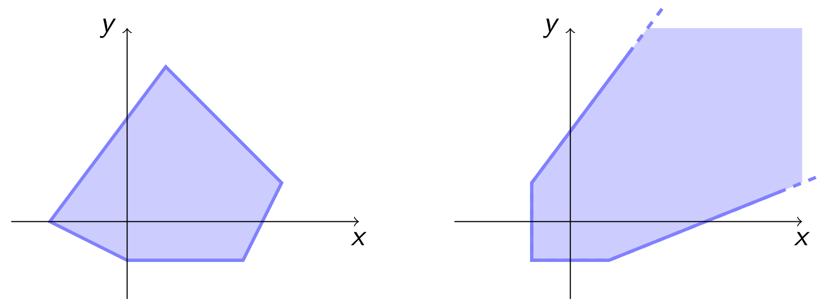


## Polyèdres

On s'intéresse aux polyèdres fermés convexes

soit des ensembles de la forme $\left\{\rho \;\middle|\; \bigwedge_i \left(\sum_j a_{ij}\rho(v_j) \geqslant b_i\right)\right\}$

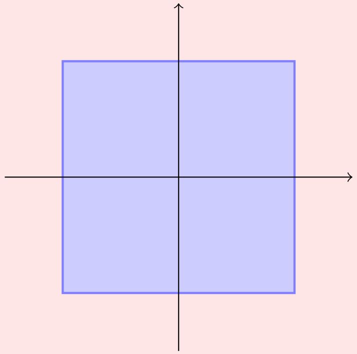
avec $a_{ij}, b_i \in \mathbb{Z}$ et $v_j \in \mathbb{V}$.



## Polyèdre, treillis

### Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.



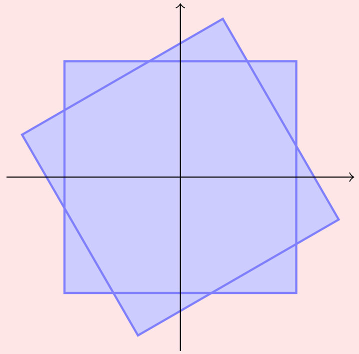
En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.



## Polyèdre, treillis

### Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.



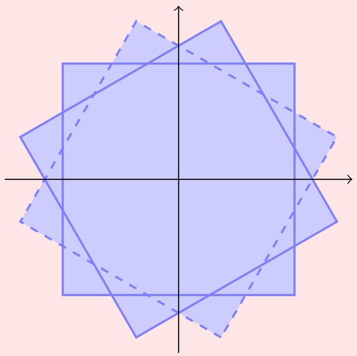
En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.



## Polyèdre, treillis

### Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.



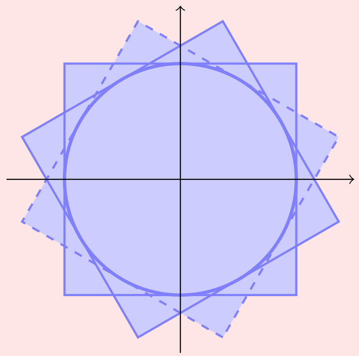
En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.



## Polyèdre, treillis

### Remarque

Les polyèdres ne forment pas un treillis :
une intersection d'une infinité de carrés peut donner un disque.

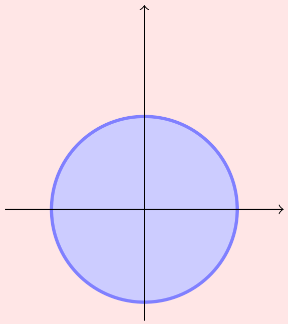


En pratique, on ne calcule que des intersections finies,
donc ce n'est pas gênant.

## Polyèdres, meilleure abstraction

**Remarque**

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.



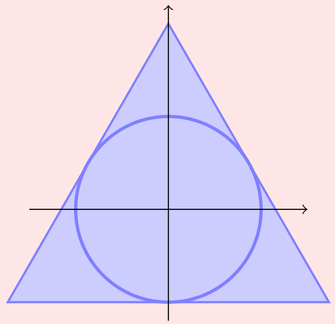
En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.



## Polyèdres, meilleure abstraction

**Remarque**

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.



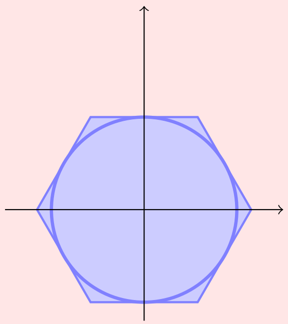
En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.



## Polyèdres, meilleure abstraction

**Remarque**

De nombreux objets concrets n'ont pas de meilleure abstraction : un disque peut être approximé par un polygone régulier à $n$ côtés, un polygone régulier à $2n$ côtés sera une meilleure abstraction.



En pratique, on ne considère que des polyèdres avec un nombre fini de côtés, donc ce n'est pas gênant.



## Représentation des polyèdres

Deux représentations duales :

**Contraintes**

$(M, c)$ avec $M \in \mathbb{Z}^{m \times n}$ et $c \in \mathbb{Z}^m$ :

$$\gamma(M, c) = \{v \mid Mv \geqslant c\}$$

avec $v = (v_1, \ldots, v_n)$ vecteur des variables ($v_i \in \mathbb{V}$).

## Représentation des polyèdres

Deux représentations duales :

### Contraintes

$(M, c)$ avec $M \in \mathbb{Z}^{m\times n}$ et $c \in \mathbb{Z}^m$ :

$$\gamma(M, c) = \{v \mid Mv \geqslant c\}$$

avec $v = (v_1, \dots, v_n)$ vecteur des variables ($v_i \in \mathbb{V}$).

### Générateurs

$(P, R)$ avec $P \in \mathbb{Z}^{n\times p}$ et $R \in \mathbb{Z}^{n\times r}$ :

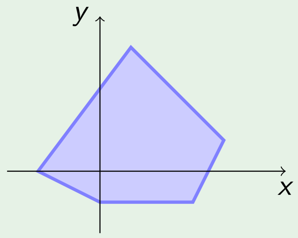
$$\gamma(P, R) = \left\{ \left(\sum_{i=1}^{p} a_i P_{.i}\right) + \left(\sum_{i=1}^{r} b_i R_{.i}\right) \;\middle|\; \begin{array}{l} \forall i, a_i \geqslant 0, b_i \geqslant 0 \\ \sum_{i=1}^{p} a_i = 1 \end{array} \right\}$$

$P$ est nommé ensemble de *points* et $R$ ensemble de *rayons*.



## Représentation des polyèdres, exemples

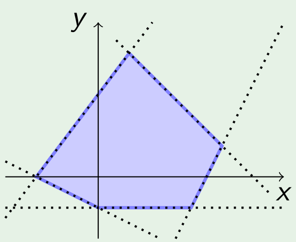
### Contraintes






## Représentation des polyèdres, exemples

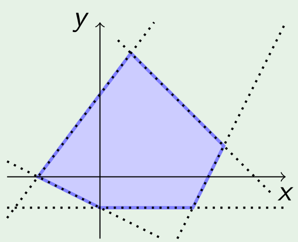
### Contraintes



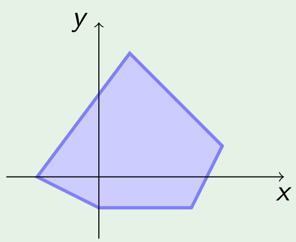



## Représentation des polyèdres, exemples

### Contraintes

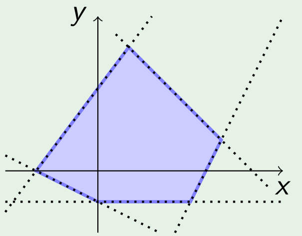



### Générateurs

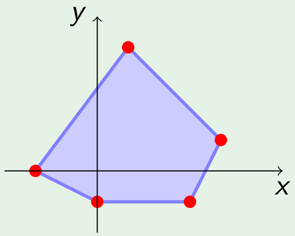


## Représentation des polyèdres, exemples

### Contraintes



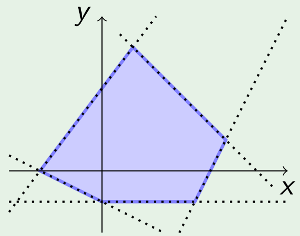

### Générateurs

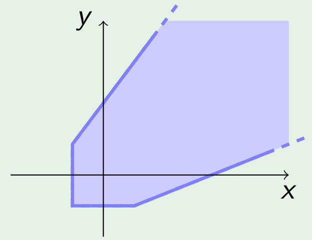




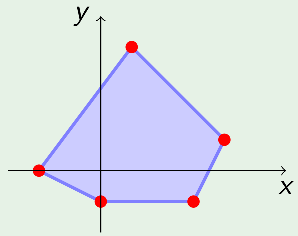

## Représentation des polyèdres, exemples

### Contraintes



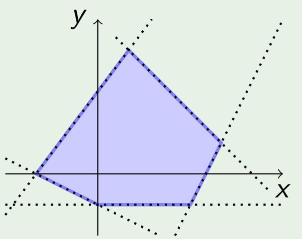

### Générateurs

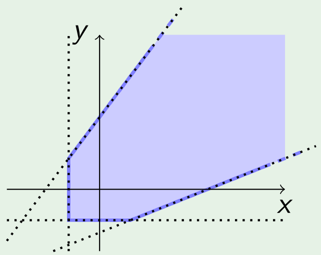




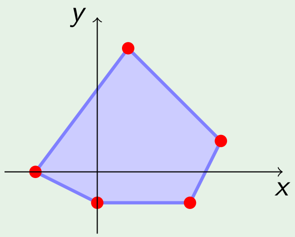

## Représentation des polyèdres, exemples

### Contraintes



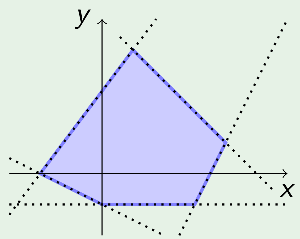

### Générateurs

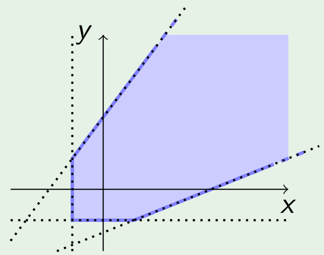




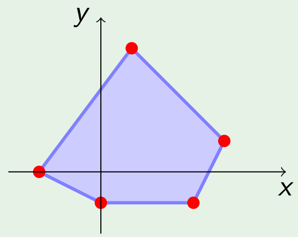

## Représentation des polyèdres, exemples

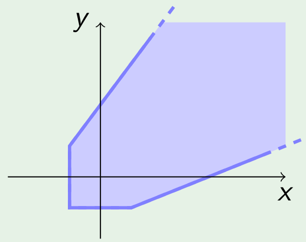
### Contraintes



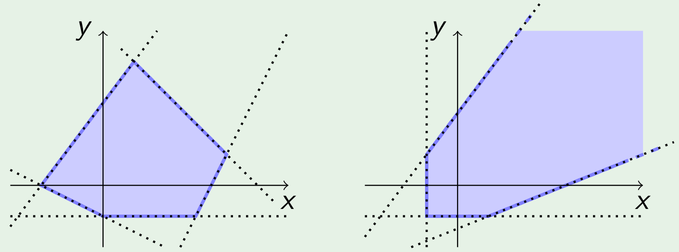

### Générateurs



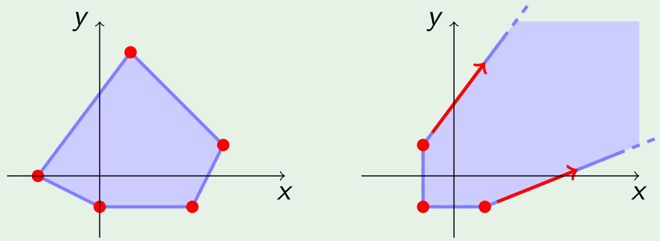



## Représentation des polyèdres, exemples

### Contraintes




### Générateurs






## Minimalité de la représentation

### Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).




## Minimalité de la représentation

### Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).


### Remarques

- La représentation minimale n'est pas unique.
  - contraintes




## Minimalité de la représentation

### Définition

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

### Remarques

- La représentation minimale n'est pas unique.
  - contraintes


  - générateurs

## Minimalité de la représentation

**Définition**

Une représentation est *minimale* si elle ne contient pas de contrainte (resp. point ou rayon) redondante (i.e. aucune contrainte (resp. point, rayon) ne peut être enlevée sans changer la concrétisation).

**Remarques**

- La représentation minimale n'est pas unique.
  - contraintes

  - générateurs

- Il est intéressant de garder une représentation minimale pour minimiser la complexité spatiale et temporelle.



## Remarques sur la dualité

**Remarques**

- Les opérations sont souvent plus faciles sur une des représentations que sur l'autre.
- On a un algorithme (Chernikova) pour passer d'une représentation à l'autre.
- Complexité au pire cas exponentielle en $n$ (l'hypercube de dimension $n$ a $2n$ faces et $2^n$ sommets).



## Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$



## Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$

## Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes



## Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes
- $x^\sharp \sqcup^\sharp y^\sharp$ : union des ensembles de générateurs



## Opérations abstraites

Grâce à la dualité, on peut calculer simplement :

- $x^\sharp \sqsubseteq^\sharp y^\sharp$ : chaque générateur de $x^\sharp$ vérifie toutes les contraintes de $y^\sharp$
- $x^\sharp =^\sharp y^\sharp$ : $x^\sharp \sqsubseteq^\sharp y^\sharp$ et $y^\sharp \sqsubseteq^\sharp x^\sharp$
- $x^\sharp \sqcap^\sharp y^\sharp$ : union des ensembles de contraintes
- $x^\sharp \sqcup^\sharp y^\sharp$ : union des ensembles de générateurs

- Gardes : on ajoute des contraintes :
$$\left[\!\left[\sum_i a_i v_i + b > 0\right]\!\right]^\sharp_{\mathrm{C}}(M, c) = \left(\begin{pmatrix} M \\ a_1 \dots a_n \end{pmatrix}, \begin{pmatrix} c \\ 1 - b \end{pmatrix}\right)$$



## Opérations abstraites, affectation

On applique simplement l'affectation aux générateurs :

$$\left[\!\left[v_i = \sum_i a_i v_i + b\right]\!\right]^\sharp_{\mathrm{C}}(P, R) = (AP + B, AR)$$

avec

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ a_1 & \cdots & a_i & \cdots & a_n \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ \vdots \\ b \\ \vdots \\ 0 \end{pmatrix}$$

## Opérations abstraites, affectation

On applique simplement l'affectation aux générateurs :

$$\left[\!\left[ v_i = \sum_i a_i v_i + b \right]\!\right]^\sharp_{\mathrm{C}} (P, R) = (AP + B, AR)$$

avec

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ a_1 & \cdots & a_i & \cdots & a_n \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ \vdots \\ b \\ \vdots \\ 0 \end{pmatrix}$$

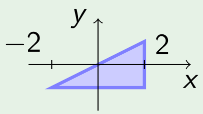
**Remarques**

- Malgré l'absence de correspondance de Galois, toutes ces opérations sont optimales (et même exactes, sauf $\sqcup^\sharp$).
- Dans le cas non linéaire, il faudrait abstraire par du linéaire...



## Opérations abstraites, affectation, exemples

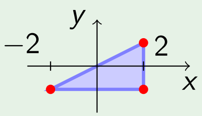
**Exemple (x = x − y − 1)**






## Opérations abstraites, affectation, exemples

**Exemple (x = x − y − 1)**

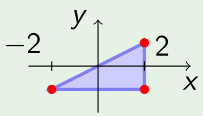



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$

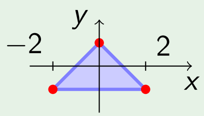


## Opérations abstraites, affectation, exemples

**Exemple (x = x − y − 1)**



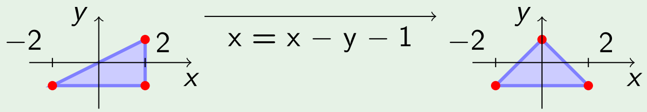

$\xrightarrow{\mathrm{x} = \mathrm{x} - \mathrm{y} - 1}$




$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$

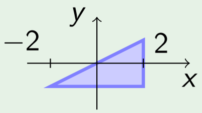
## Opérations abstraites, affectation, exemples

### Exemple (x = x − y − 1)




$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$

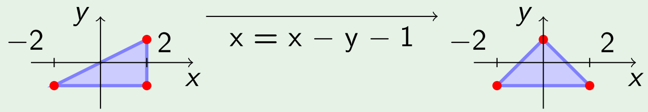
### Exemple (x = 2y)





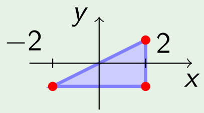

## Opérations abstraites, affectation, exemples

### Exemple (x = x − y − 1)



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$

### Exemple (x = 2y)

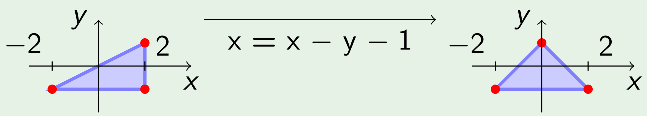


$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$



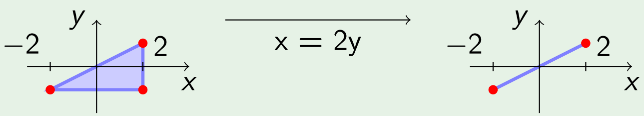
## Opérations abstraites, affectation, exemples

### Exemple (x = x − y − 1)



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$

### Exemple (x = 2y)

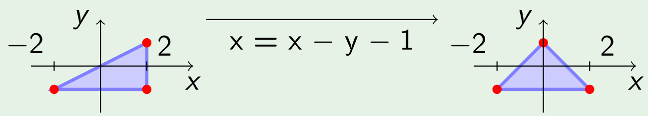



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & -2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$



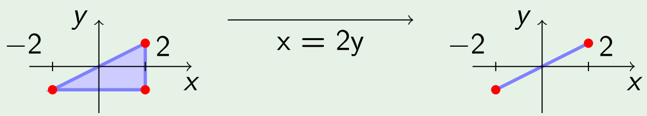
## Opérations abstraites, affectation, exemples

### Exemple (x = x − y − 1)



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} -1 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & 2 & 0 \\ -1 & -1 & 1 \end{pmatrix}$

### Exemple (x = 2y)



$P = \begin{pmatrix} -2 & 2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$ $\begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix} P + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 & -2 & 2 \\ -1 & -1 & 1 \end{pmatrix}$

### Exercice (*)

Définir l'opérateur abstrait d'affectation sur les contraintes.

## Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).



## Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).

### Idée

Toujours la même : ne conserver que les contraintes stables.



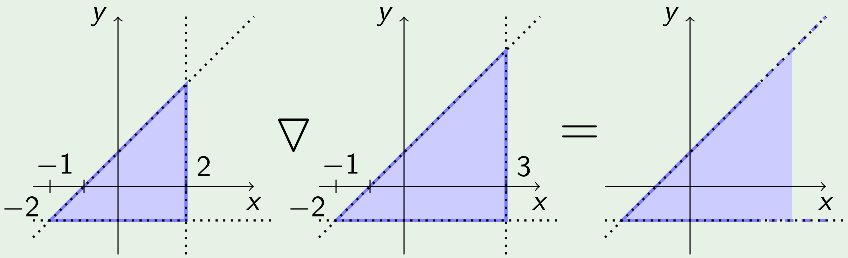
## Élargissement

On a des chaînes croissantes infinies
donc il nous faut un élargissement (widening).

### Idée

Toujours la même : ne conserver que les contraintes stables.

### Exemple





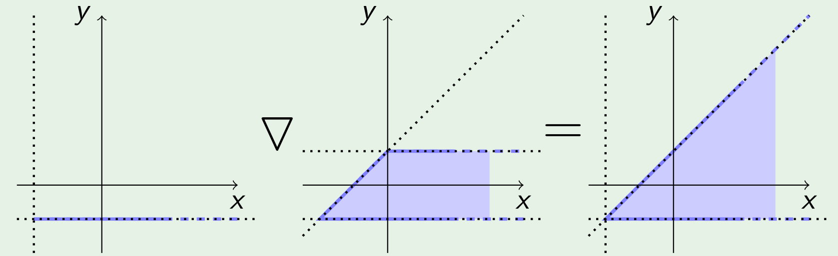

## Élargissement (suite et fin)

Plus formellement :

### Définition

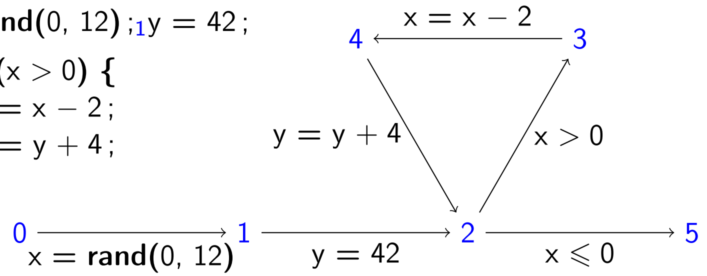
Pour $x^\sharp$ et $y^\sharp$ sous forme d'ensemble de contraintes minimaux,
$x^\sharp \nabla y^\sharp =$
$\left\{ c \in x^\sharp \;\middle|\; y^\sharp \in \{\, c \,\} \right\} \cup \left\{ c \in y^\sharp \;\middle|\; \exists c' \in x^\sharp, x^\sharp =^\sharp \left( x^\sharp \setminus c' \right) \cup \{\, c \,\} \right\}$

### Exemple



## Exemple

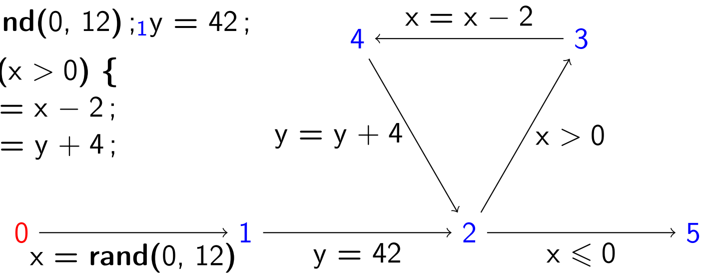
$_0x = \textbf{rand}(0,\ 12)\ ;_1y = 42\ ;$
**while** $_2(x > 0)$ **{**
$\quad _3x = x - 2\ ;$
$\quad _4y = y + 4\ ;$
**}**$_5$






## Exemple

$_0x = \textbf{rand}(0,\ 12)\ ;_1y = 42\ ;$
**while** $_2(x > 0)$ **{**
$\quad _3x = x - 2\ ;$
$\quad _4y = y + 4\ ;$
**}**$_5$



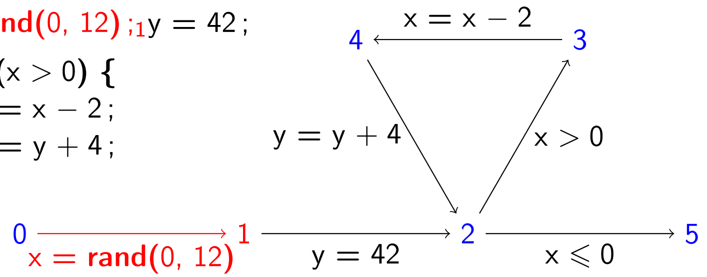

$\top$

0

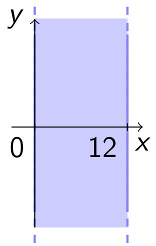


## Exemple

$_0x = \textbf{rand}(0,\ 12)\ ;_1y = 42\ ;$
**while** $_2(x > 0)$ **{**
$\quad _3x = x - 2\ ;$
$\quad _4y = y + 4\ ;$
**}**$_5$




$\top$

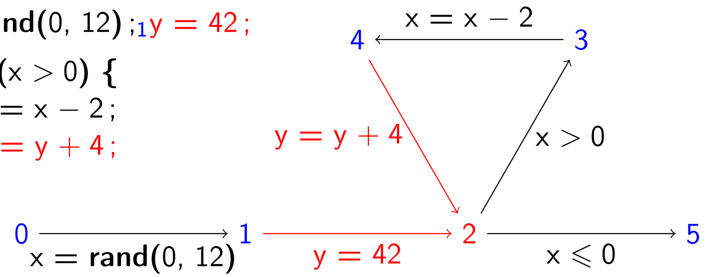



0 1

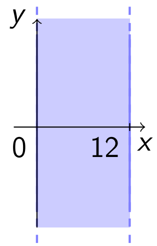


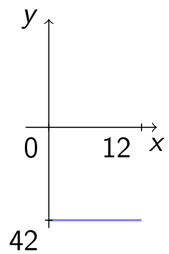
## Exemple

$_0x = \textbf{rand}(0,\ 12)\ ;_1y = 42\ ;$
**while** $_2(x > 0)$ **{**
$\quad _3x = x - 2\ ;$
$\quad _4y = y + 4\ ;$
**}**$_5$




$\top$

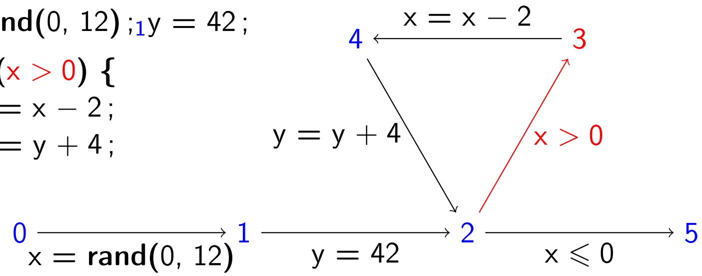


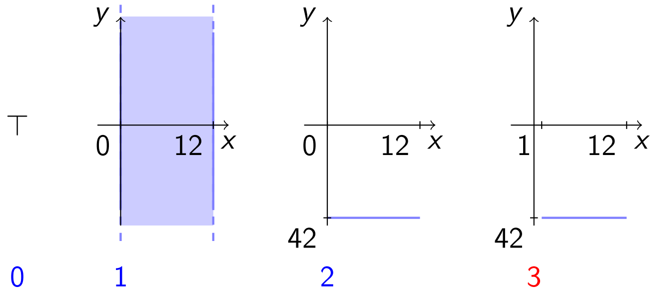




0 1 2

## Exemple

$_0\mathrm{x} = \mathbf{rand}(0,\ 12)\ ;_1\mathrm{y} = 42\ ;$
$\mathbf{while}\ _2(\mathrm{x} > 0)\ \{$
$\quad _3\mathrm{x} = \mathrm{x} - 2\ ;$
$\quad _4\mathrm{y} = \mathrm{y} + 4\ ;$
$\}_5$

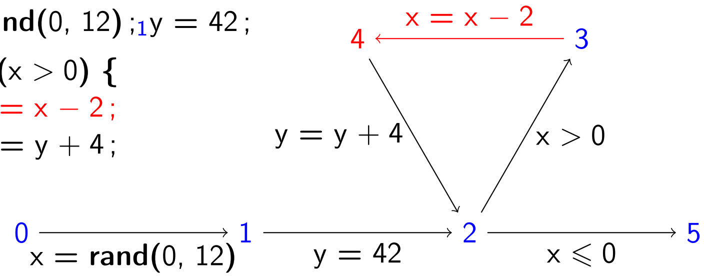


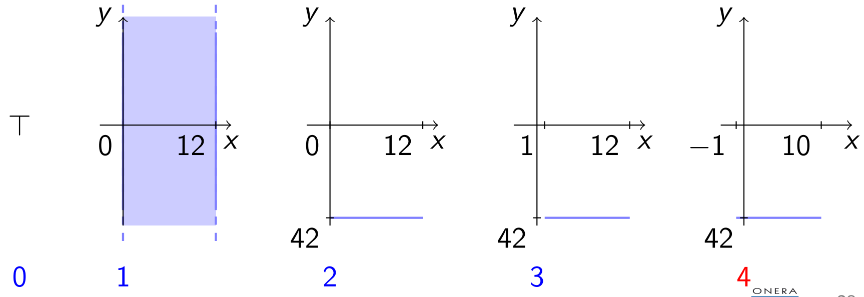






## Exemple

$_0\mathrm{x} = \mathbf{rand}(0,\ 12)\ ;_1\mathrm{y} = 42\ ;$
$\mathbf{while}\ _2(\mathrm{x} > 0)\ \{$
$\quad _3\mathrm{x} = \mathrm{x} - 2\ ;$
$\quad _4\mathrm{y} = \mathrm{y} + 4\ ;$
$\}_5$

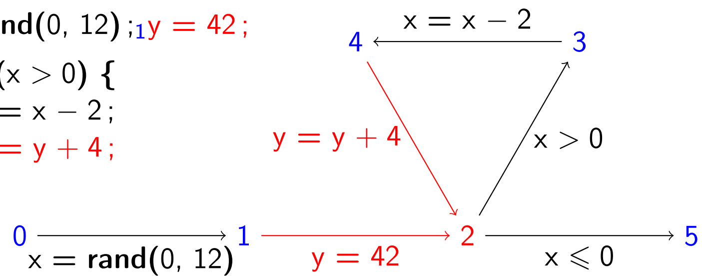


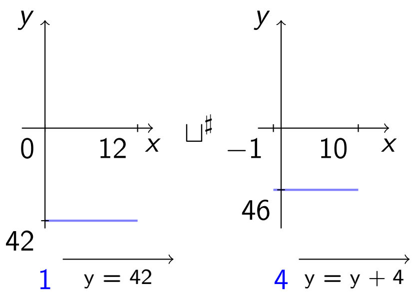






## Exemple (suite)

$_0\mathrm{x} = \mathbf{rand}(0,\ 12)\ ;_1\mathrm{y} = 42\ ;$
$\mathbf{while}\ _2(\mathrm{x} > 0)\ \{$
$\quad _3\mathrm{x} = \mathrm{x} - 2\ ;$
$\quad _4\mathrm{y} = \mathrm{y} + 4\ ;$
$\}_5$

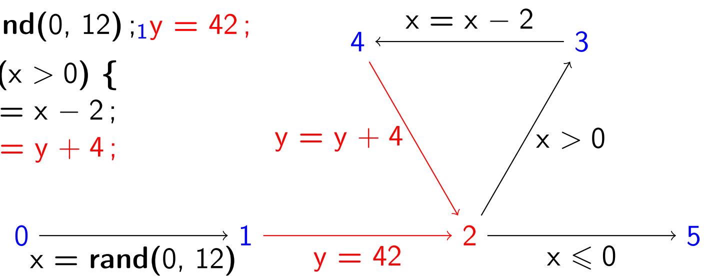


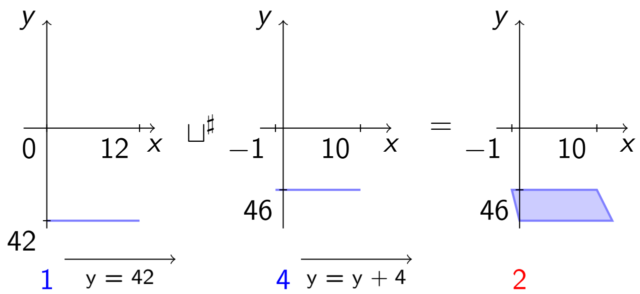






## Exemple (suite)

$_0\mathrm{x} = \mathbf{rand}(0,\ 12)\ ;_1\mathrm{y} = 42\ ;$
$\mathbf{while}\ _2(\mathrm{x} > 0)\ \{$
$\quad _3\mathrm{x} = \mathrm{x} - 2\ ;$
$\quad _4\mathrm{y} = \mathrm{y} + 4\ ;$
$\}_5$



## Exemple (suite)

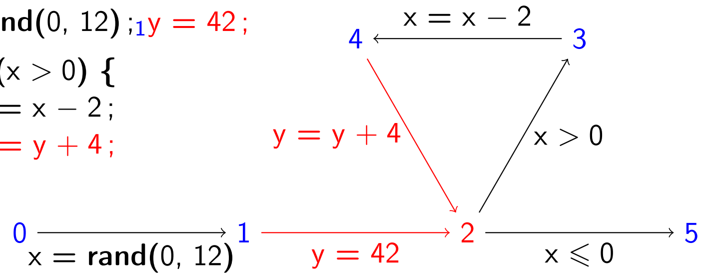
${}_0\mathtt{x} = \mathbf{rand}(0,\ 12)\ ;{}_1\mathtt{y} = 42\ ;$

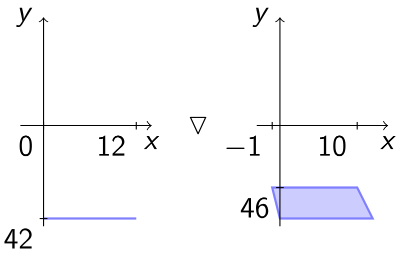
$\mathbf{while}\ {}_2(\mathtt{x} > 0)\ \{$

$\quad {}_3\mathtt{x} = \mathtt{x} - 2\ ;$

$\quad {}_4\mathtt{y} = \mathtt{y} + 4\ ;$

$\}_5$






## Exemple (suite)

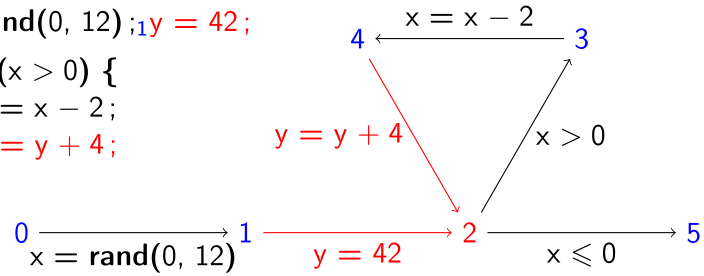
${}_0\mathtt{x} = \mathbf{rand}(0,\ 12)\ ;{}_1\mathtt{y} = 42\ ;$

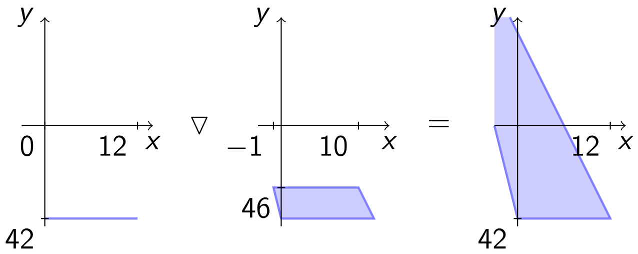
$\mathbf{while}\ {}_2(\mathtt{x} > 0)\ \{$

$\quad {}_3\mathtt{x} = \mathtt{x} - 2\ ;$

$\quad {}_4\mathtt{y} = \mathtt{y} + 4\ ;$

$\}_5$




2



## Exemple (suite)

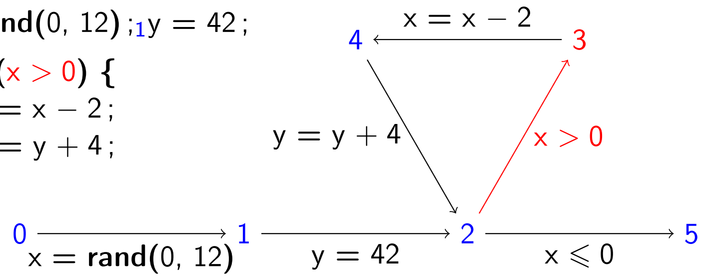
${}_0\mathtt{x} = \mathbf{rand}(0,\ 12)\ ;{}_1\mathtt{y} = 42\ ;$

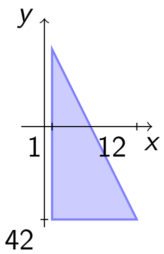
$\mathbf{while}\ {}_2(\mathtt{x} > 0)\ \{$

$\quad {}_3\mathtt{x} = \mathtt{x} - 2\ ;$

$\quad {}_4\mathtt{y} = \mathtt{y} + 4\ ;$

$\}_5$




3



## Exemple (suite)

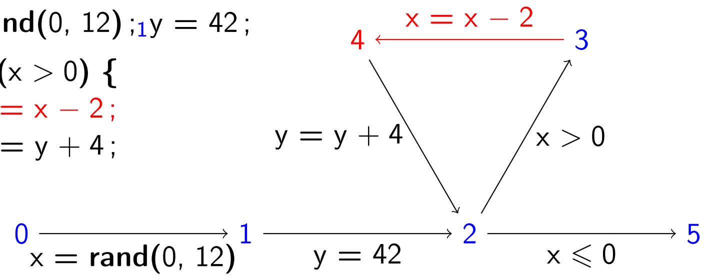
${}_0\mathtt{x} = \mathbf{rand}(0,\ 12)\ ;{}_1\mathtt{y} = 42\ ;$

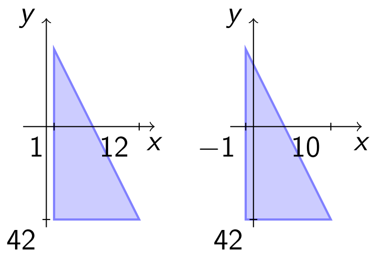
$\mathbf{while}\ {}_2(\mathtt{x} > 0)\ \{$

$\quad {}_3\mathtt{x} = \mathtt{x} - 2\ ;$

$\quad {}_4\mathtt{y} = \mathtt{y} + 4\ ;$

$\}_5$




3 4

## Exemple (suite)

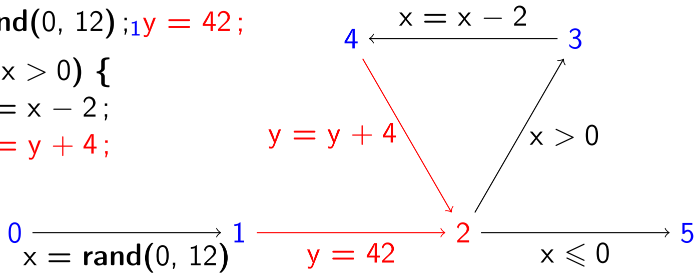
$_0x = \mathbf{rand}(0, 12)\,;\,_1y = 42\,;$

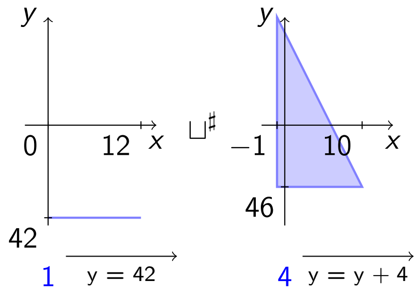
**while** $_2(x > 0)$ {

  $_3x = x - 2\,;$

  $_4y = y + 4\,;$

}$_5$






## Exemple (suite)

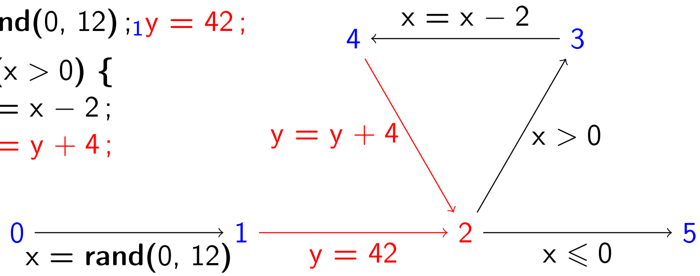
$_0x = \mathbf{rand}(0, 12)\,;\,_1y = 42\,;$

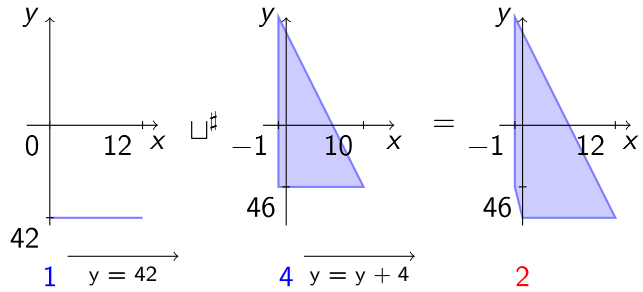
**while** $_2(x > 0)$ {

  $_3x = x - 2\,;$

  $_4y = y + 4\,;$

}$_5$






## Exemple (suite)

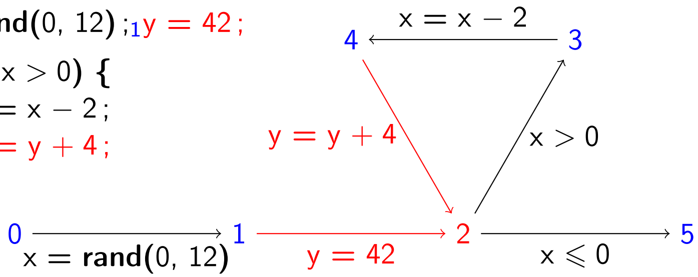
$_0x = \mathbf{rand}(0, 12)\,;\,_1y = 42\,;$

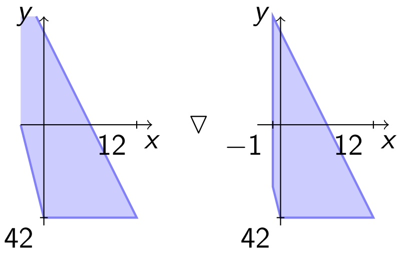
**while** $_2(x > 0)$ {

  $_3x = x - 2\,;$

  $_4y = y + 4\,;$

}$_5$






## Exemple (suite)

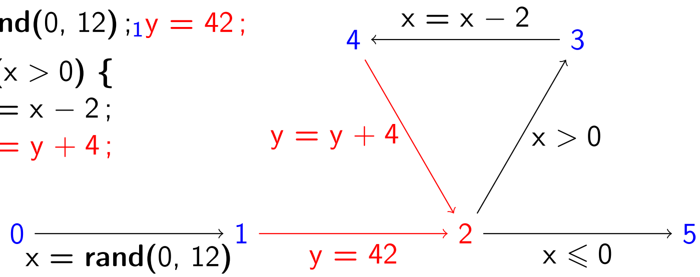
$_0x = \mathbf{rand}(0, 12)\,;\,_1y = 42\,;$

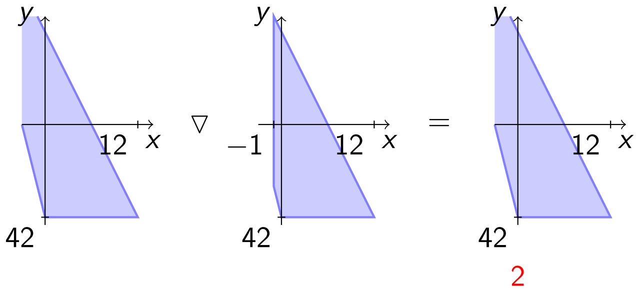
**while** $_2(x > 0)$ {

  $_3x = x - 2\,;$

  $_4y = y + 4\,;$

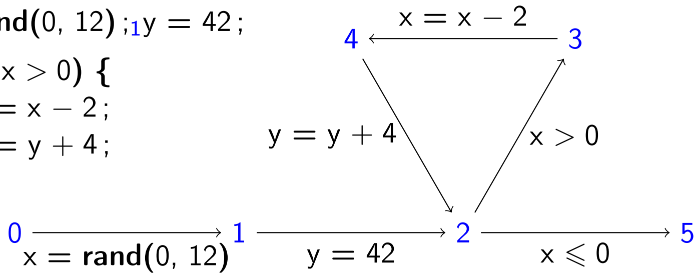
}$_5$

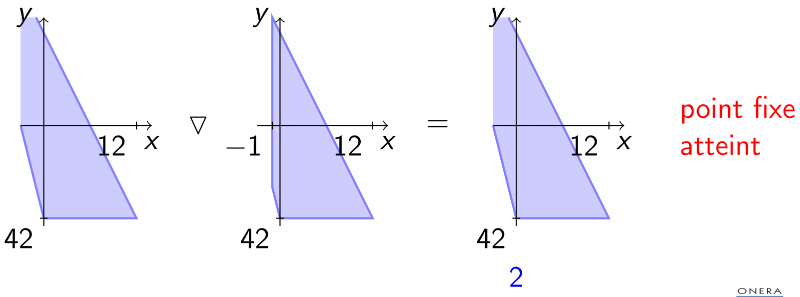


## Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) **{**
  $_3$x = x − 2 ;
  $_4$y = y + 4 ;
**}**$_5$

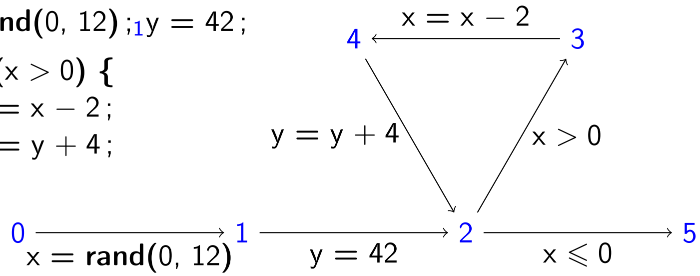


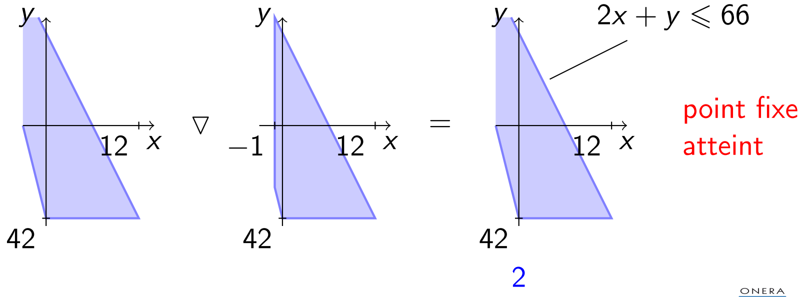






## Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) **{**
  $_3$x = x − 2 ;
  $_4$y = y + 4 ;
**}**$_5$

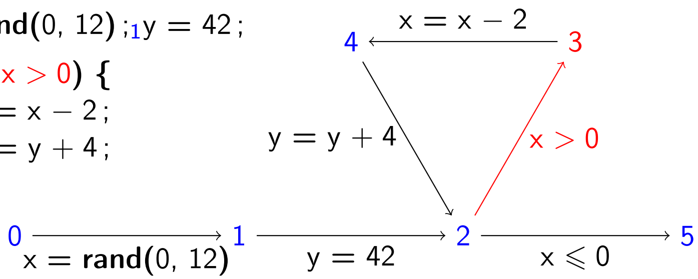


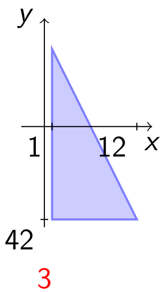






## Exemple (suite)

$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) **{**
  $_3$x = x − 2 ;
  $_4$y = y + 4 ;
**}**$_5$

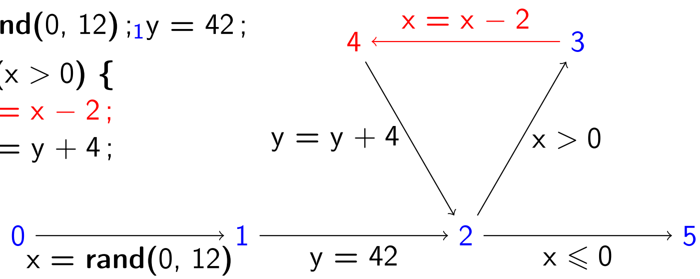


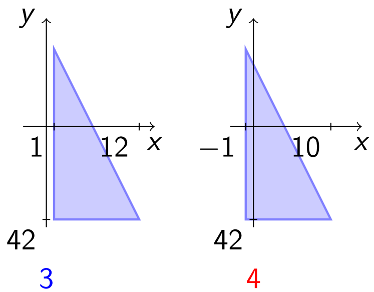






## Exemple (suite)

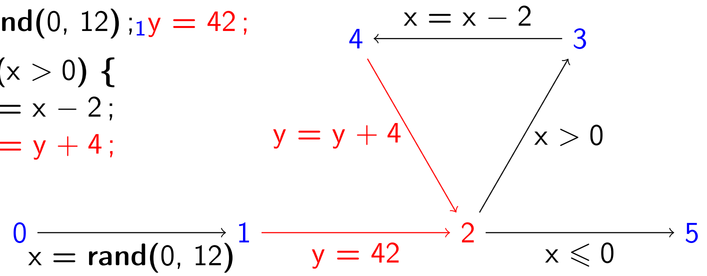
$_0$x = **rand**(0, 12) ;$_1$y = 42 ;
**while** $_2$(x > 0) **{**
  $_3$x = x − 2 ;
  $_4$y = y + 4 ;
**}**$_5$

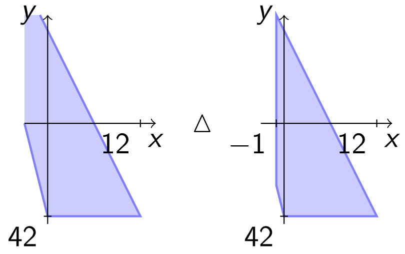


## Exemple (suite et fin)

${}_0$x = **rand**(0, 12) ; ${}_1$y = 42 ;
**while** ${}_2$(x > 0) {
  ${}_3$x = x − 2 ;
  ${}_4$y = y + 4 ;
}${}_5$

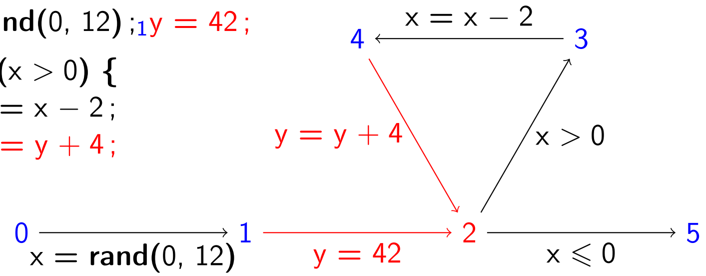


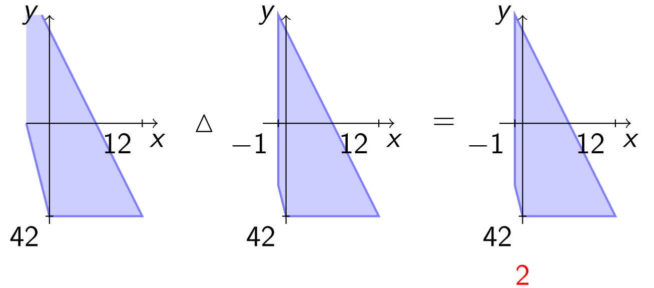






## Exemple (suite et fin)

${}_0$x = **rand**(0, 12) ; ${}_1$y = 42 ;
**while** ${}_2$(x > 0) {
  ${}_3$x = x − 2 ;
  ${}_4$y = y + 4 ;
}${}_5$

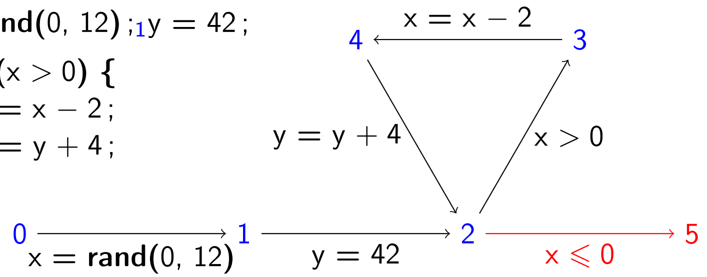


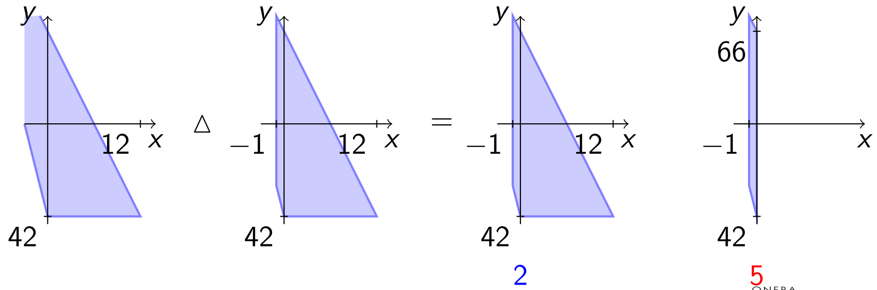






## Exemple (suite et fin)

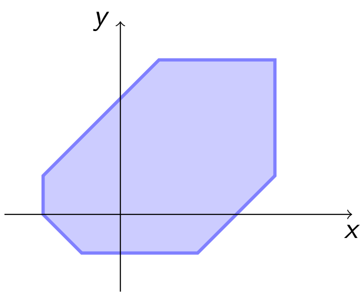
${}_0$x = **rand**(0, 12) ; ${}_1$y = 42 ;
**while** ${}_2$(x > 0) {
  ${}_3$x = x − 2 ;
  ${}_4$y = y + 4 ;
}${}_5$

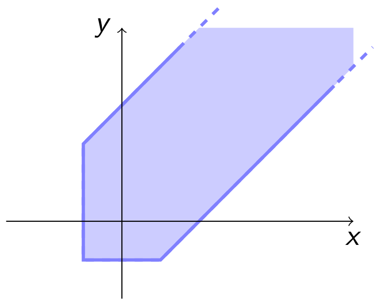


## Octogones

Similaire aux polyèdres mais en autorisant seulement les pentes multiples de 45°.



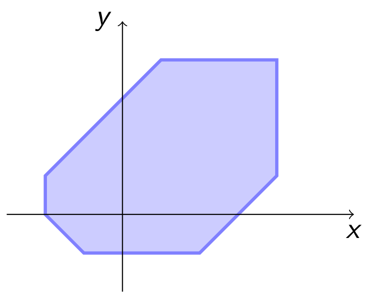



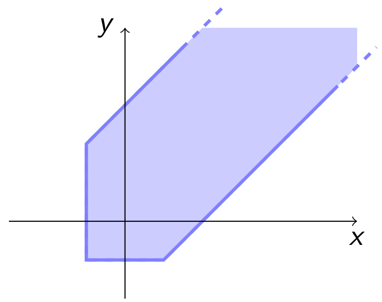




## Octogones

Similaire aux polyèdres mais en autorisant seulement les pentes multiples de 45°.


− moins précis

\+ meilleure complexité : chaque opération est en $\mathrm{O}\left(n^3\right)$ (complexité au pire cas exponentielle pour les polyèdres)



## Octogones, exercice

**Exercice**

On considère le programme suivant :

$_0$x = **rand**(0, 12) ;$_1$y = 0 ;

**while** $_2$(x > 0) **{**

  $_3$**if** (**rand**(0, 1) > 0) **{**

    $_4$x = x − 1 ;

  **}** **else** **{**

    $_5$x = x − 2 ;

  **}**

  $_6$y = y + 1 ;

**}**$_7$

1. Dessiner le graphe de flot de contrôle.
2. Calculer le point fixe.
3. Le raffiner par une itération descendante (avec $\triangle$).



## Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

## Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

- égalités affines ($2x + 3y = 5$)



## Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

- égalités affines ($2x + 3y = 5$)
- congruences ($x + 2y$ congru à 3 modulo 5)

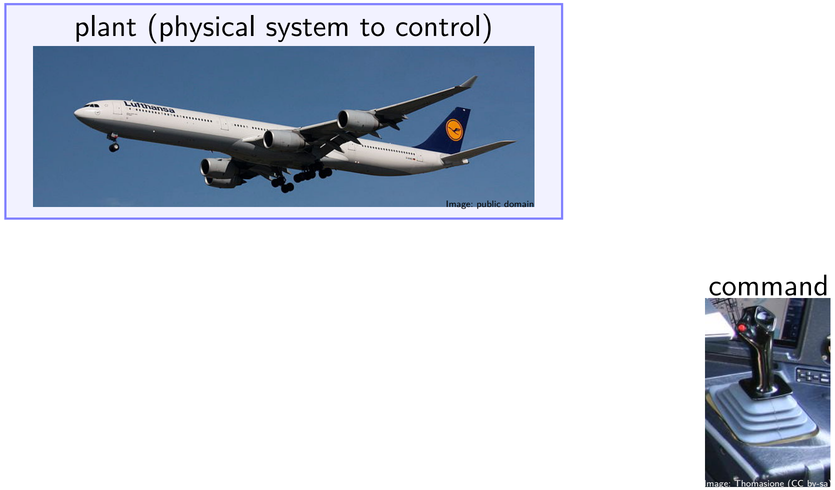


## Autres domaines relationnels

Il existe bien d'autres domaines relationnels :

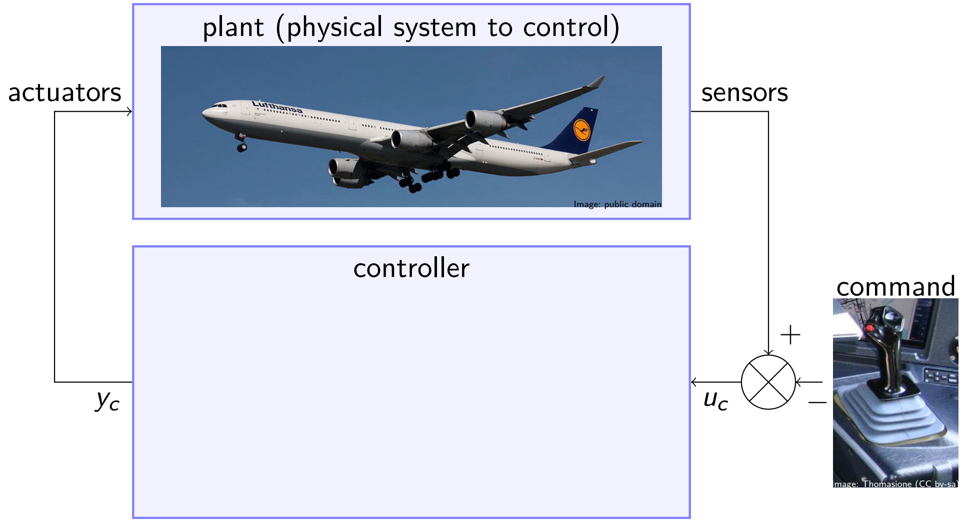
- égalités affines ($2x + 3y = 5$)
- congruences ($x + 2y$ congru à 3 modulo 5)
- polyèdres tropicaux (polyèdres sur une algèbre $(\max, +)$)
- ...

# Systèmes de control-commande



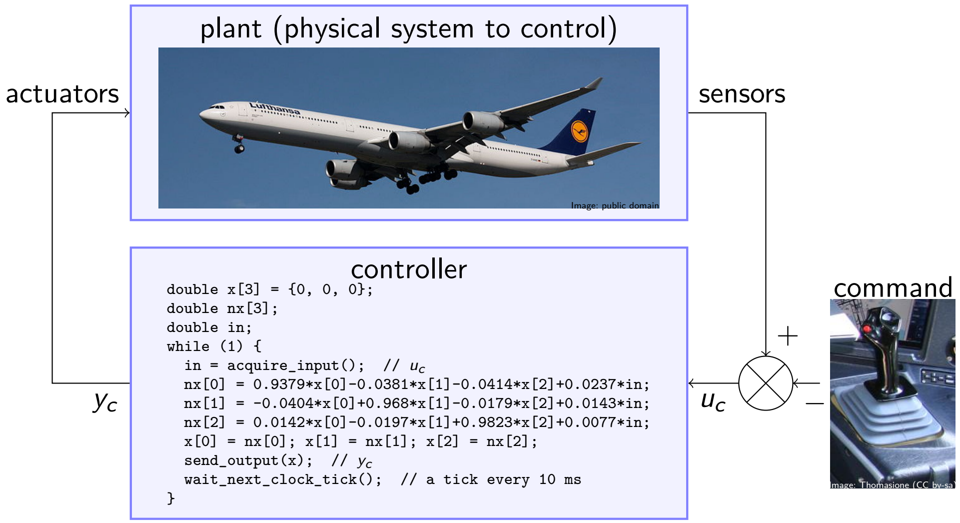



# Systèmes de control-commande





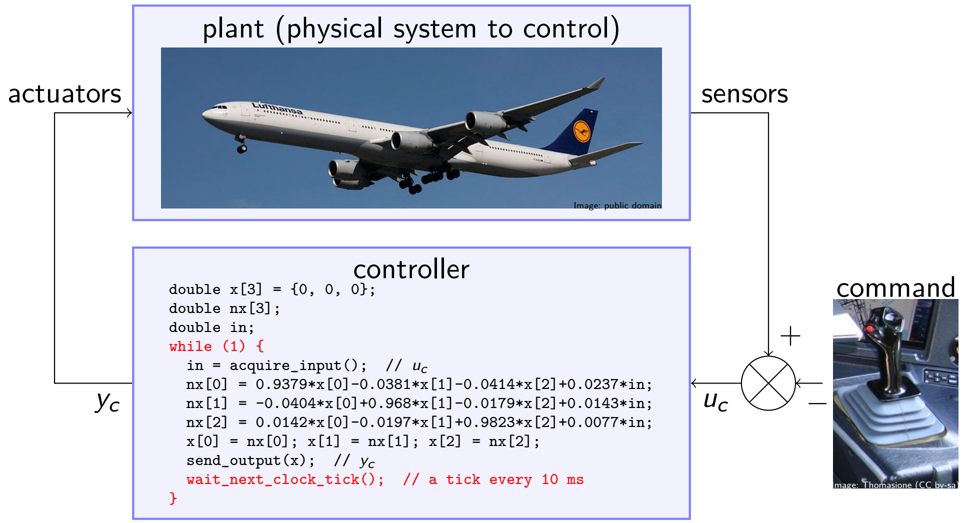

# Systèmes de control-commande




```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```



# Systèmes de control-commande




```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

## Systèmes de control-commande

plant (physical system to control)


actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

command


$y_c$ $u_c$ $+$ $-$



## Systèmes de control-commande

plant (physical system to control)

actuators

sensors

controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

command

$y_c$ $u_c$ $+$ $-$

ONERA 41 / 68

## Systèmes de control-commande

plant (physical system to control)

actuators

sensors

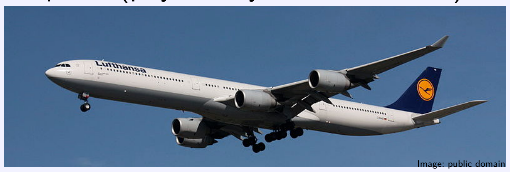
controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

command

$y_c$ $u_c$ $+$ $-$

ONERA 41 / 68



## Systèmes de control-commande

plant (physical system to control)

actuators

sensors

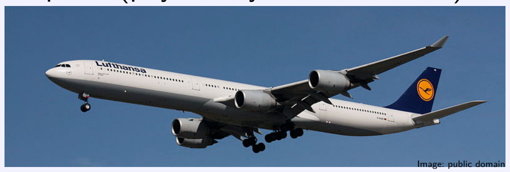
controller

```
double x[3] = {0, 0, 0};
double nx[3];
double in;
while (1) {
  in = acquire_input();  // u_c
  nx[0] = 0.9379*x[0]-0.0381*x[1]-0.0414*x[2]+0.0237*in;
  nx[1] = -0.0404*x[0]+0.968*x[1]-0.0179*x[2]+0.0143*in;
  nx[2] = 0.0142*x[0]-0.0197*x[1]+0.9823*x[2]+0.0077*in;
  x[0] = nx[0]; x[1] = nx[1]; x[2] = nx[2];
  send_output(x);  // y_c
  wait_next_clock_tick();  // a tick every 10 ms
}
```

command

$y_c$ $u_c$ $+$ $-$

ONERA 41 / 68



# Étude des propriétés

Propriétés :

- du système complet : système + controleur - boucle fermée
- du controleur seul - boucle ouverte

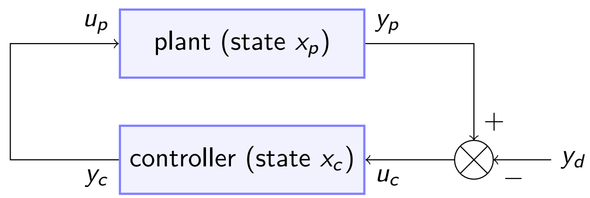
Propriétés classiques :

- stabilité de la boucle fermée, de la boucle ouverte
- robustesse
- performance (overshoot borné, temps pour atteindre la consigne, ...)



# Stabilité

- Stabilité de la boucle fermée



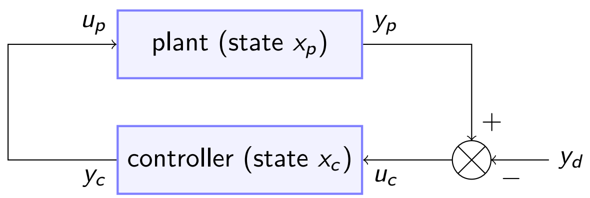

commande $y_d$ bornée $\Rightarrow$ $x_c$ et $x_p$ bornés

(et donc $y_c$ et $y_p$ bornés)



# Stabilité

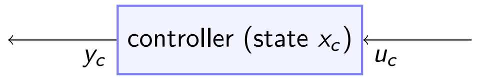
- Stabilité de la boucle fermée




commande $y_d$ bornée $\Rightarrow$ $x_c$ et $x_p$ bornés

(et donc $y_c$ et $y_p$ bornés)

- Stabilité de la boucle ouverte




input $u_c$ borné $\Rightarrow$ $x_c$ borné (et donc $y_c$ borné)



# Analyse statique de controleurs

Le contrôleur

```
x0 := 0 ; x1 := 0 ; x2 := 0 ;
while −1 ⩽ 0 do
  in := ?(−1, 1) ;
  x0' := x0 ; x1' := x1 ; x2' := x2 ;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in ;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in ;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in ;
od
```

est stable en boucle ouverte :

$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251.$

## Analyse statique de controleurs

Le contrôleur

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ⩽ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

est stable en boucle ouverte :
$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251$.

**Objectif**

Construire un programme (*d'analyse statique*)
pour calculer ces bornes à partir du code source.



## Analyse statique de controleurs

Le contrôleur

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ⩽ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

est stable en boucle ouverte :
$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251$.

**Objectif**

Construire un programme (*d'analyse statique*)
pour calculer ces bornes à partir du code source.



## Analyse statique de controleurs

Le contrôleur

```
x0 := 0; x1 := 0; x2 := 0;
while −1 ⩽ 0 do
  in := ?(−1, 1);
  x0' := x0; x1' := x1; x2' := x2;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in;
od
```

est stable en boucle ouverte :
$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251$.

**Objectif**

Construire un programme (*d'analyse statique*)
pour calculer ces bornes à partir du code source.

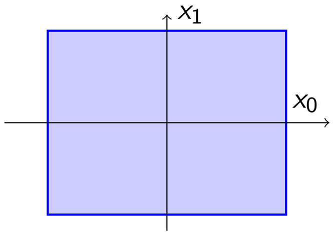


## Types d'invariants

- les *invariants linéaires* utilisés habituellement en analyse statique ne sont pas adaptés :
  - au mieux, ils sont coûteux ;
  - au pire, inefficaces.

## Types d'invariants

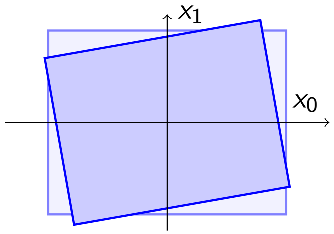
- les *invariants linéaires* utilisés habituellement en analyse statique ne sont pas adaptés :
  - au mieux, ils sont coûteux ;
  - au pire, inefficaces.






## Types d'invariants

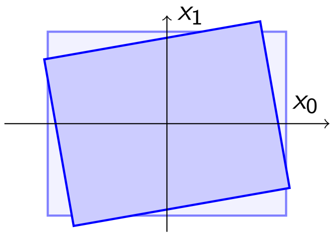
- les *invariants linéaires* utilisés habituellement en analyse statique ne sont pas adaptés :
  - au mieux, ils sont coûteux ;
  - au pire, inefficaces.

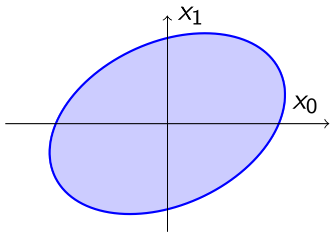





## Types d'invariants

- les *invariants linéaires* utilisés habituellement en analyse statique ne sont pas adaptés :
  - au mieux, ils sont coûteux ;
  - au pire, inefficaces.
- les automaticiens savent depuis longtemps que les *invairants quadratiques*
  sont pertinents pour l'analyse de systèmes linéaires.

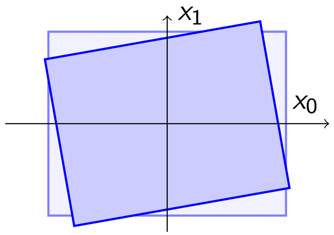


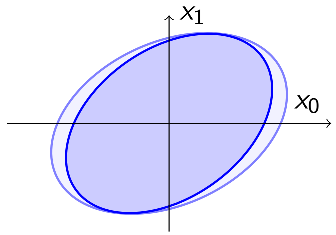






## Types d'invariants

- les *invariants linéaires* utilisés habituellement en analyse statique ne sont pas adaptés :
  - au mieux, ils sont coûteux ;
  - au pire, inefficaces.
- les automaticiens savent depuis longtemps que les *invairants quadratiques*
  sont pertinents pour l'analyse de systèmes linéaires.



## Invariants quadratiques

**Remark**

L'espace d'état réel n'est *pas* en général un ellipsoïde.



## Invariants quadratiques

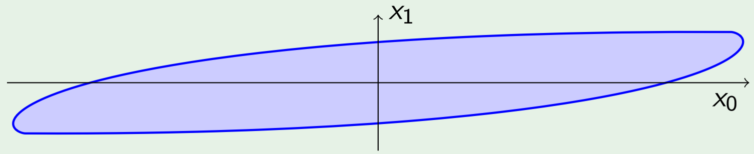
**Remark**

L'espace d'état réel n'est *pas* en général un ellipsoïde.

**Exemple**

$x_0 := 0$ et $x_{k+1} := Ax_k + Bu_k$ où $\|u_k\|_\infty \leqslant 1$ et

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$






## Invariants quadratiques

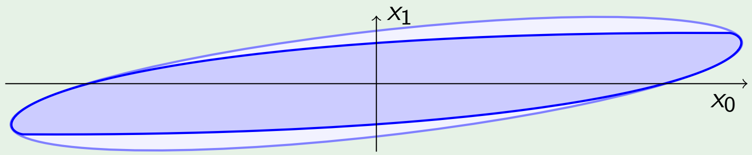
**Remark**

L'espace d'état réel n'est *pas* en général un ellipsoïde.

**Exemple**

$x_0 := 0$ et $x_{k+1} := Ax_k + Bu_k$ où $\|u_k\|_\infty \leqslant 1$ et

$$A := \begin{bmatrix} 0.92565 & -0.0935 \\ 0.00935 & 0.935 \end{bmatrix} \qquad B := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$




Mais c'est pas loin.



## Stabilité de Lyapunov [Lyapunov47]

**Theoreme**

Pour tout $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times p}$, la série

$$\begin{cases} x_0 \in \mathbb{R}^n \\ x_{k+1} = Ax_k + Bu_k \end{cases}$$

est bornée pour tout $u \in (\mathbb{R}^p)^{\mathbb{N}}$ tel que pour tout $k \in \mathbb{N}$, $||u_k||_\infty \leqslant 1$ ssi il existe $P \in \mathbb{R}^{n \times n}$ semi-définie positif tel que

$$P - A^T P A \succ 0$$

où $M \succ 0$ signifie que pour tout $x \in \mathbb{R}^n : x \neq 0 \Rightarrow x^T M x > 0$.

## Stabilité de Lyapunov, Invariant

**Ellipsoide invariante**

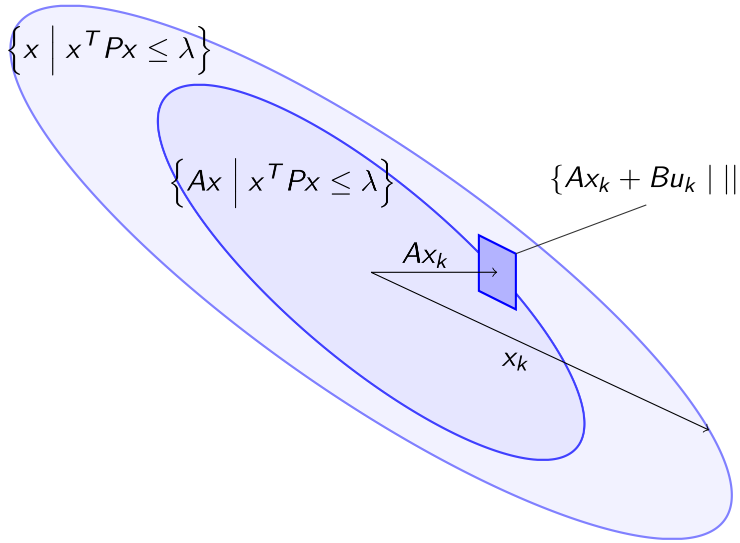
De plus, il existe un $\lambda > 0$ tel que
$x$ reste dans l'ellipsoide $\left\{x \in \mathbb{R}^n \,\middle|\, x^T P\, x \leqslant \lambda\right\}$.

**Pour un informaticien**

La propriété "$x^T P\, x \leqslant \lambda$" est un *invariant de boucle*.



## Stabilité de Lyapunov, Illustration






## Outils

Pour résoudre l'équation de Lyapunov $P - A^T P\, A \succ 0$ :

**Semidefinite Programming [VandenbergheB96]**

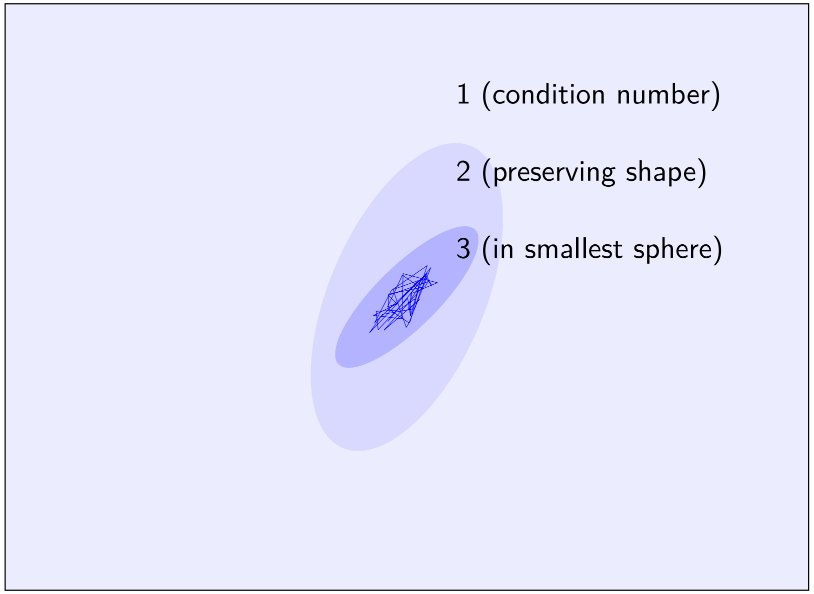
Minimise un fonction objectif linéaire en les variables $y_i$
sous la contrainte

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

où les matrices $A_i$ sont connues
et $M \succeq 0$ signifie $x^T M\, x \geqslant 0$ pour tout vecteur $x$.



## PLusieurs solutions



# Calcul de la borne

Sur le code

x0 := 0 ; x1 := 0 ; x2 := 0 ;
**while** $-1 \leqslant 0$ **do**
  in := **?**$(-1, 1)$ ;
  x0' := x0 ; x1' := x1 ; x2' := x2 ;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in ;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in ;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in ;
**od**

l'outil calcule la forme quadratique puis les bornes

$6.2547x_0^2 + 12.1868x_1^2 + 3.8775x_2^2 - 10.61x_0x_1 - 2.4306x_0x_2 + 2.4182x_1x_2 \leqslant 1.0029$

$\wedge x_0^2 \leqslant 0.1795 \wedge x_1^2 \leqslant 0.1136 \wedge x_2^2 \leqslant 0.2757$

enfin

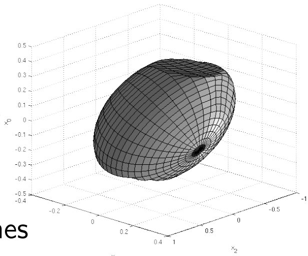
$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251.$



# Calcul de la borne

Sur le code

x0 := 0 ; x1 := 0 ; x2 := 0 ;
**while** $-1 \leqslant 0$ **do**
  in := **?**$(-1, 1)$ ;
  x0' := x0 ; x1' := x1 ; x2' := x2 ;
  x0 := 0.9379×x0'−0.0381×x1'−0.0414×x2'+0.0237×in ;
  x1 := −0.0404×x0'+0.968×x1'−0.0179×x2'+0.0143×in ;
  x2 := 0.0142×x0'−0.0197×x1'+0.9823×x2'+0.0077×in ;
**od**



l'outil calcule la forme quadratique puis les bornes

$6.2547x_0^2 + 12.1868x_1^2 + 3.8775x_2^2 - 10.61x_0x_1 - 2.4306x_0x_2 + 2.4182x_1x_2 \leqslant 1.0029$

$\wedge x_0^2 \leqslant 0.1795 \wedge x_1^2 \leqslant 0.1136 \wedge x_2^2 \leqslant 0.2757$

enfin

$|x_0| \leqslant 0.4236 \wedge |x_1| \leqslant 0.3371 \wedge |x_2| \leqslant 0.5251.$



Sémantique abstraite - suite

Abstractions relationnelles
- Rappel
- Polyèdres
- Octogones

Contrôleurs d'avion

Si le cours avait duré un semestre...
- Domaines non numériques
- Virgule flottante
- Partitionnement
- Stratégies d'itération

Outils existants



Sémantique abstraite - suite

Abstractions relationnelles
- Rappel
- Polyèdres
- Octogones

Contrôleurs d'avion

Si le cours avait duré un semestre...
- Domaines non numériques
- Virgule flottante
- Partitionnement
- Stratégies d'itération

Outils existants

## Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

### Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).



## Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

### Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).

Exemple : concaténation de deux listes

| @ | nil | non_nil |
|---|---|---|
| nil | nil | non_nil |
| non_nil | non_nil | non_nil |



## Domaines non numériques

Tous les domaines abstraits ne sont pas numériques.

### Exemple (listes)

On peut abstraire une liste en retenant
si elle est vide (nil) ou non (non_nil).

Exemple : concaténation de deux listes

| @ | nil | non_nil |
|---|---|---|
| nil | nil | non_nil |
| non_nil | non_nil | non_nil |

Exemple d'utilisation : prouver qu'on n'essaye jamais d'acceder
à la tête d'une liste vide (`List.hd []` en Caml).

## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.



## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.



## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).



## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

Solutions :

- pour les intervalles : arrondir les bornes vers l'extérieur ;



## Virgule flottante

- Les nombres réels ne sont pas représentable en machine.
- On utilise donc des nombres à virgule flottante.
- D'où des erreurs d'arrondi (démo).
- Problème : comment abstraire correctement ces arrondis.

Solutions :

- pour les intervalles : arrondir les bornes vers l'extérieur ;
- plus généralement : on peut abstraire une opération flottante $\mathrm{round}(a + b)$ par une opération réelle $(1 + \epsilon)(a + b)$ puis utiliser des domaines sur les réels ;
- reste alors à implémenter correctement des domaines sur les réels, c'est un autre problème (on peut utiliser des rationnels par exemple).







## Partitionnement, exemple

$_0$x = **rand(**-12, 12**)** ;<br>
$_1$**if** (x > 0) **{**<br>
    $_2$x = x + 1 ;<br>
**} else {**<br>
    $_3$x = x − 1 ;<br>
**}**<br>
$_4$y = 1 / x ;$_5$

## Partitionnement, exemple

$_{0}x = \textbf{rand}(\text{-}12,\ 12)\,;$

$_{1}\textbf{if}\ (x > 0)\ \{$

$\quad _{2}x = x + 1\,;$

$\}\ \textbf{else}\ \{$

$\quad _{3}x = x - 1\,;$

$\}$

$_{4}y = 1\ /\ x\,;_{5}$

- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$



## Partitionnement, exemple

$_{0}x = \textbf{rand}(\text{-}12,\ 12)\,;$

$_{1}\textbf{if}\ (x > 0)\ \{$

$\quad _{2}x = x + 1\,;$

$\}\ \textbf{else}\ \{$

$\quad _{3}x = x - 1\,;$

$\}$

$_{4}y = 1\ /\ x\,;_{5}$

- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$
- D'où en 4, $x \in [\![-13, 13]\!]$
  et une fausse alarme division par 0



## Partitionnement, exemple

$_{0}x = \textbf{rand}(\text{-}12,\ 12)\,;$

$_{1}\textbf{if}\ (x > 0)\ \{$

$\quad _{2}x = x + 1\,;$

$\}\ \textbf{else}\ \{$

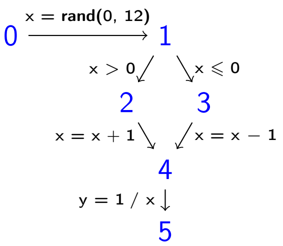
$\quad _{3}x = x - 1\,;$

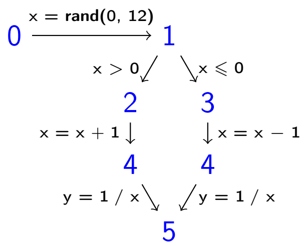
$\}$

$_{4}y = 1\ /\ x\,;_{5}$

- Après 2, on a a $x \in [\![2, 13]\!]$
- Après 3, on a a $x \in [\![-13, -1]\!]$
- D'où en 4, $x \in [\![-13, 13]\!]$
  et une fausse alarme division par 0

Solution : déplacer le calcul de la borne supérieure des intervalles après l'affectation y := 1 / x.



## Stratégies d'itération

- Le widening/narrowing marche plutôt bien.



## Stratégies d'itération

- Le widening/narrowing marche plutôt bien.
- Mais il est difficile de concevoir un bon widening.



## Stratégies d'itération

- Le widening/narrowing marche plutôt bien.
- Mais il est difficile de concevoir un bon widening.
- D'où l'intérêt pour d'autres méthodes d'itération :
  - accélération ;
  - itération sur les stratégies (policy iteration).

Sémantique abstraite - suite

Abstractions relationnelles
- Rappel
- Polyèdres
- Octogones

Contrôleurs d'avion

Si le cours avait duré un semestre...
- Domaines non numériques
- Virgule flottante
- Partitionnement
- Stratégies d'itération

Outils existants



## Astrée

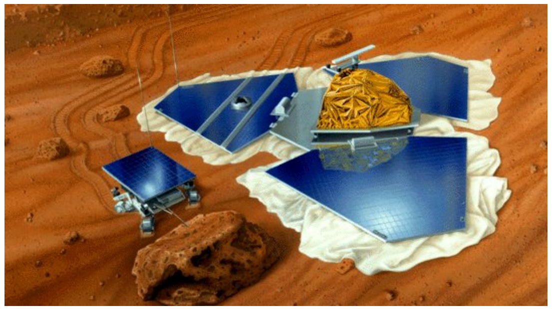
- Développé par l'équipe de Patrick Cousot à l'ÉNS Ulm.
- Preuve d'absence d'erreur à l'exécution dans du code temps réel embarqué.
- Utilisé pour les commandes de vol des Airbus (plusieurs centaines de milliers de lignes de C).


http://www.astree.ens.fr/



## IKOS – Inference Kernel for Static Analyzers

- Développé par la NASA. Open-source
- Objectifs similaires à Astrée.
- successeur de CGS – C Global Surveyor
- CGS - Utilisé sur les contrôleurs de vols de : Mars Pathfinder, Deep Space One,. . .

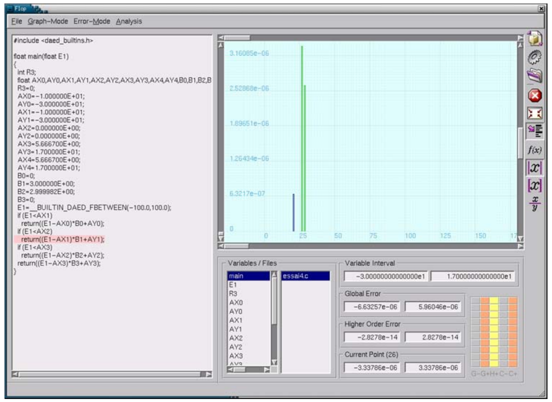


http://ti.arc.nasa.gov/opensource/ikos/



## Fluctuat

- Développé par l'équipe d'Éric Goubault au CEA.
- Analyse des erreurs d'arrondi en virgule flottante.
- Utilisé par divers industriels.



http://www-list.cea.fr/labos/fr/LSL/fluctuat/index.html

## Polyspace

- Vendu par MathWorks.
- Plus généraliste.
- Moins précis.
- Utilisé par divers industriels.

http://www.polyspace.com/



## Apron

- Librairie de domaines relationnels
  développée par Bertrand Jeannet (INRIA Rhône-Alpes)
  et Antoine Miné (CNRS, ÉNS).
- Polyèdres.
- Octogones.
- Implémenté en C.
- Interface en OCaml.

http://apron.cri.ensmp.fr/library/